UNIVERSIDADE DE SÃO PAULO

ICMC - USP

Jefferson Yudi Ihida, 8922368

Matheus Cabral Manoel, 9066470

Pedro Renan Goulart, 8531783

"Simulador Universal de Autômatos Finitos"

SÃO CARLOS

2020

SUMÁRIO

INTRODUÇÃO……..……..……..2
TÉCNICAS UTILIZADAS..…….....……....……..……..2
QUALIDADE.…….....……....……..……..4
COMO RODAR..…….....……....……..……..4
CONCLUSÃO…….....……....……..……..4

INTRODUÇÃO

Neste trabalho, nossa equipe desenvolveu um simulador universal de autômatos finitos. Autômatos finitos são dispositivos que julgam a legalidade de cadeias sobre algum alfabeto finito. Para isso, leremos um arquivo de entrada que define o autômato finito e algumas cadeias de entrada. Nosso simulador processará as cadeias no autômato definido, printando a aceitação ou não de cada uma, na ordem em que foram processadas.

TÉCNICAS UTILIZADAS

Para a implementação, utilizamos a linguagem python, já que todos os integrantes do grupo tem alguma familiaridade com ela.

O algoritmo implementado, em linhas gerais, é o seguinte:

```
26 if __name__ == '__main__':
27     entrada = le_entrada()
28     automato = entrada_para_estrutura()
29     if eh_deterministico(automato):
30         avalia_automato_deterministico(automato)
31     else
32         avalia_automato_deterministico(
33             transforma_em_deterministico(automato)
34         )
```

Primeiro, lemos a entrada crua. Em seguida, passamos ela para um formato que nos permite navegar o grafo de estado de forma mais fácil. A partir disso, podemos saber se o autômato é determinístico ou não determinístico. Caso seja, nós processamos avaliamos a aceitação da cadeia. Caso não seja, nós transformamos em um determinístico equivalente e só então processamos e avaliamos a aceitação da cadeia.

O algoritmo foi definido junto com o esqueleto das subtarefas necessárias para resolução do problema:

```python
# Lê entrada e retorna string
def le_entrada():
    pass


# Pega string de entrada e retorna estrutura definida
def entrada_para_estrutura(entrada):
    pass


# Recebe estrutura e retorna true se é determinístico e false se é não determinístico
def eh_deterministicO(automato):
    pass


# Recebe estrutura de automato deterministico e printa aceito ou não aceito
def avalia_automato_deterministico(automato):
    pass


# Recebe um automato nao deterministico e retorna um automato deterministico equivalente
def transforma_em_deterministico(automato_nao_deterministico):
    pass
```

Isso foi uma parte de nosso modo de trabalho, fundamental para garantir nosso sucesso em meio à situação adversa de pandemia que impossibilitou reuniões físicas do trabalho. O modo de trabalho completo está descrito a seguir:

1. Discussão do problema.
2. Discussão de possíveis soluções e escolha de solução.
3. Implementação do algoritmo.
4. Implementação do esqueleto inicial.
5. Divisão de tarefas.
6. Programação concomitante de cada tarefa em chamada por voz conjunta.

Utilizamos um repositório no github para coordenar as implementações de diferentes subtarefas de forma concomitante.

QUALIDADE

A primeira questão importante a ser comentada é nossa decisão em realizar a conversão de um autômato finito não determinístico (AFN) para um autômato finito determinístico (AFD). Tal conversão é fundamental, já que em um AFN existe um conjunto de estados iniciais e um conjunto de próximos estados possíveis. Consequentemente, uma cadeia pode induzir caminhos múltiplos através de um AFN, com alguns terminando em estado de aceitação e outros não. Tal ambiguidade implicaria na necessidade de processar todos os caminhos para avaliar a aceitação de uma cadeia. Converter uma AFN para AFD, portanto, significa um aumento de eficiência de tempo a depender do algoritmo de conversão, além de não precisarmos mudar a forma de avaliação das cadeias, já que elas são diferentes .

A estruturação do código está bem compartimentalizada, com funções responsáveis por cada subtarefa. Isso facilita a manutenção e o entendimento de quem está o utilizando.

A eficiência do processamento das cadeias é de O(n^2), sendo n o número de caracteres da cadeia. A eficiência da conversão é de O(n^3).

COMO RODAR

A entrada lida deve estar em um arquivo chamado “entrada.txt” na mesma pasta do executável. O resultado será escrito em um arquivo chamado “saída.txt”.

Apenas mude o arquivo.txt para a entrada que preferir e abra o executável.

CONCLUSÃO

A criação de um simulador universal de autômatos finitos foi concluída. Os testes realizados foram um sucesso. O desafio foi importante para vermos na prática conceitos teóricos aprendidos em aula.